


# Redictado Taller de programación 2021

## CLASE 2

Conceptos teóricos de la Recursión

A silver laptop is placed on a light-colored wooden desk. The laptop screen is open and displays a block of Pascal code. The background is a plain, light-colored wall.

```
Program HolaMundo;  
Begin  
  writeln('Hola mundo');  
end.
```

# Temas de la clase

- Recursión. Concepto. Motivación
- Ejemplo de recursión
- ¿Cómo funciona la recursión?
- Características de un algoritmo recursivo
- Ejercitación

# Recursión

La recursión es una metodología para resolver problemas.

Permite resolver un problema **P** por resolución de instancias más pequeñas **P<sub>1</sub>**, **P<sub>2</sub>**, ..., **P<sub>n</sub>** del mismo problema.

El problema **P<sub>i</sub>** es de la misma naturaleza que el problema original, pero en algún sentido es más simple.



**Veamos un ejemplo ...**

# ***Recursión***

## **Problema:**

Buscar una palabra en un diccionario que tiene 2000 páginas.

## **Solución:**

Paso 1)

Abrir el diccionario a la mitad



**Sofi**



# *Recursión*

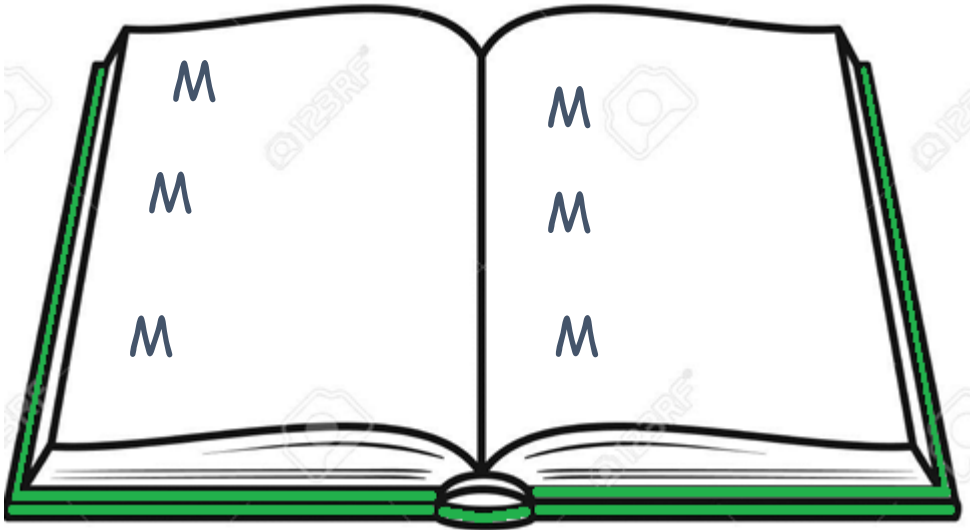
## **Problema:**

Buscar una palabra en un diccionario que tiene 2000 páginas.

## **Solución:**

Paso 2)

Quedarse con la mitad donde está la palabra





# *Recursión*

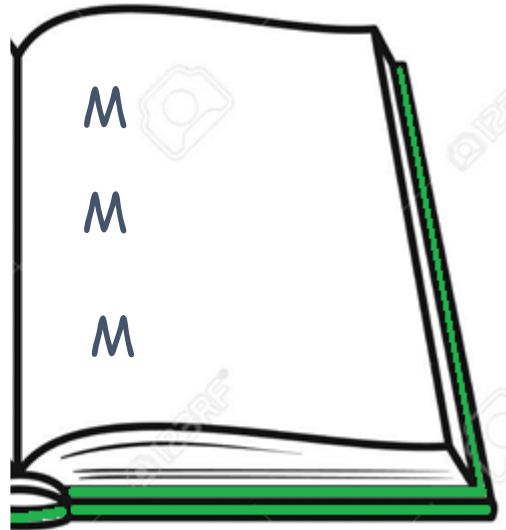
## **Problema:**

Buscar una palabra en un diccionario que tiene 1000 páginas.

## **Solución:**

Paso 1)

Abrir el diccionario a la mitad





Sofi



# *Recursión*

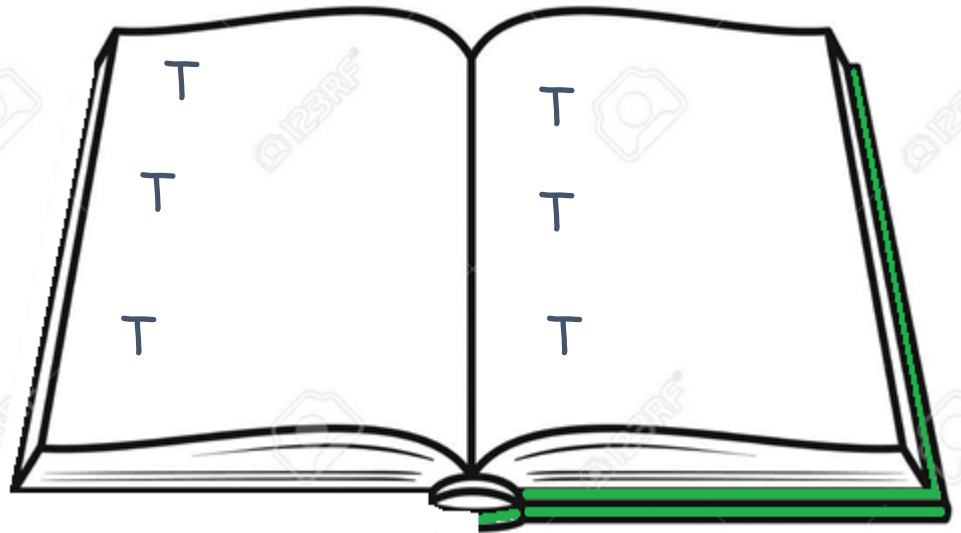
## **Problema:**

Buscar una palabra en un diccionario que tiene 1000 páginas.

## **Solución:**

Paso 2)

Quedarse con la mitad donde está la palabra





# *Recursión*

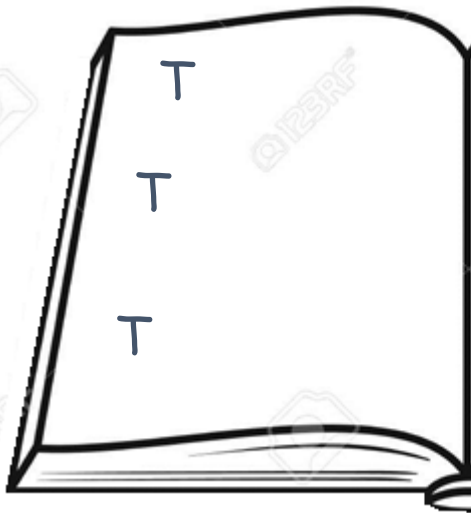
## **Problema:**

Buscar una palabra en un diccionario que tiene 500 páginas.

## **Solución:**

Paso 1

Paso 2





# *Solución recursiva*



**Sofi**

## **Caso recursivo:**

- ❖ El problema es siempre el mismo, pero en cierto sentido es cada vez más pequeño.
- ❖ La tarea a realizar es siempre la misma y en cierto sentido consiste en reducir el espacio donde hacer la búsqueda.

## **Problema:**

Buscar una palabra en un diccionario.

## **Solución:**

Paso 1) Abrir el diccionario a la mitad.

Paso 2) Quedarse con la mitad donde está la palabra.

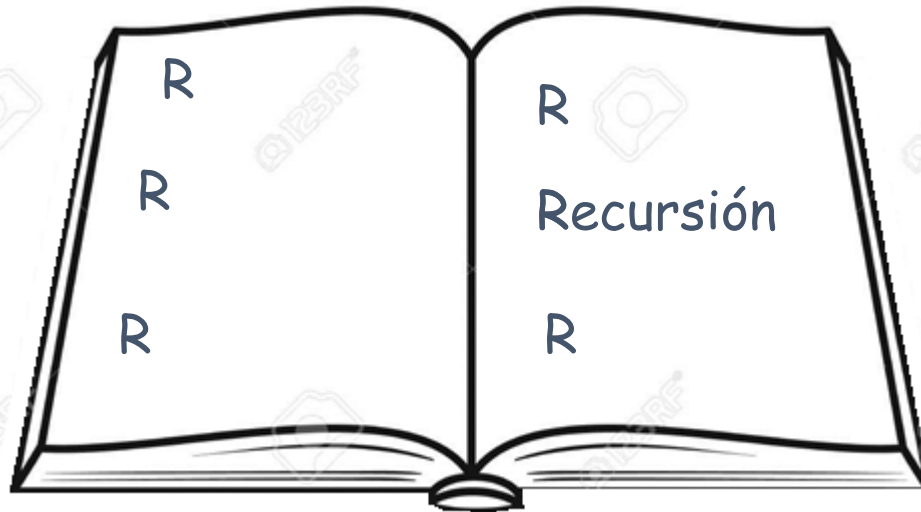


Sofi

# *Recursión*

**¿Cuándo termino?**

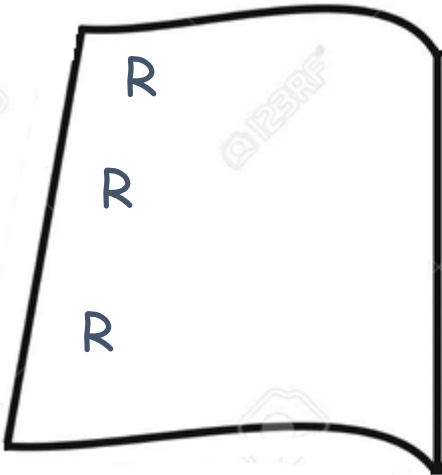
a) En algún momento abrirás el diccionario a la mitad y encontraras la palabra.



# *Recursión*



Sofi



**¿Cuándo  
termino?**

b) Si ya no puedes seguir partiendo a la mitad el diccionario, la palabra buscada no está.

# *Solución recursiva*



**Sofi**

## **Caso base:**

- ❖ Es una instancia del problema donde no se puede seguir dividiendo el problema
- ❖ El caso base termina con la recursión

**¿Cuándo termino?**

- a) Se encuentra la palabra.
- b) La palabra buscada no existe.

# Recursión

## Características de un algoritmo recursivo

Una **solución recursiva** resuelve un problema por resolución de instancias más pequeñas del mismo problema.

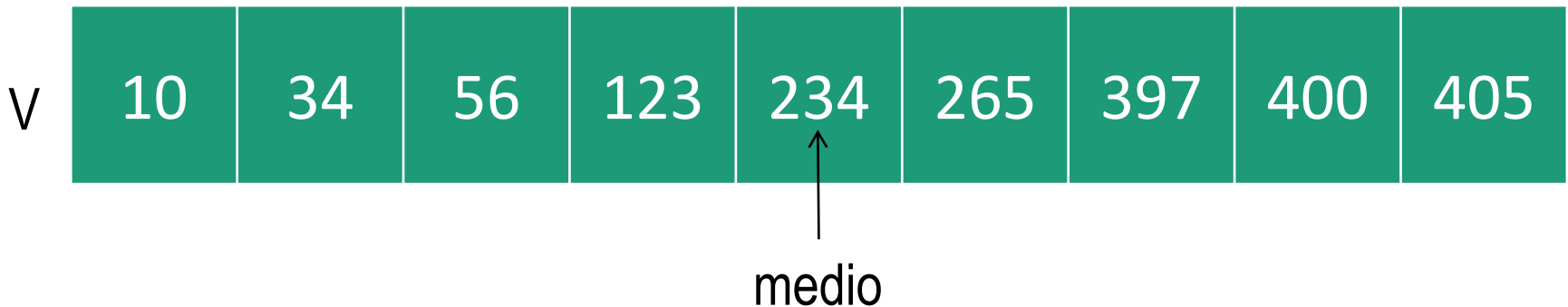
Un algoritmo recursivo involucra:

- al menos una condición de terminación (implícita / explícita) (Caso base)
- al menos una *autoinvocación* (Caso recursivo). Se debe garantizar que en un número finito de *autoinvocaciones* se alcanza la condición de terminación

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Buscar el valor 56 en el vector



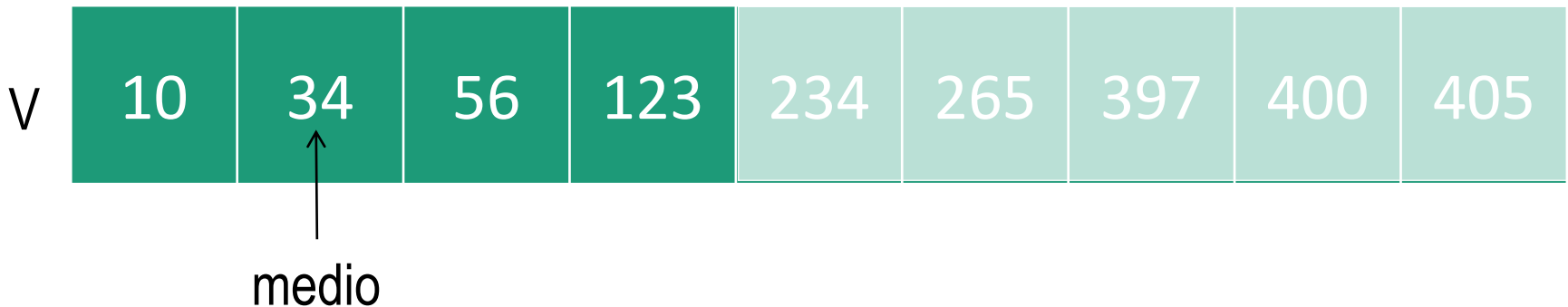
¿Cómo es 56 con respecto a  $v[\text{medio}]$ ?

1. Si es = terminé
2. Si es < busco en la mitad inferior
3. Si es > busco en la mitad superior

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Buscar el valor 56 en el vector



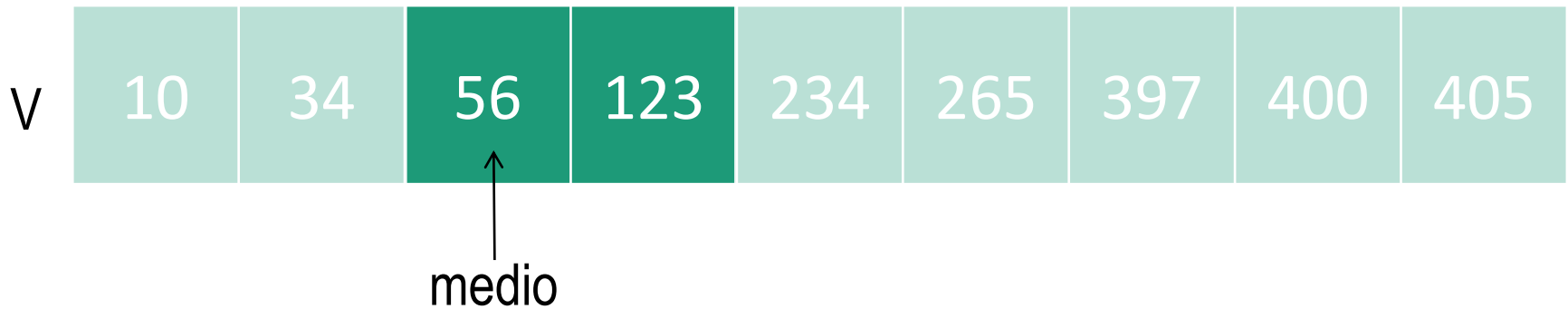
¿Cómo es 56 con respecto a  $v[\text{medio}]$ ?

1. Si es = terminé
2. Si es < busco en la mitad inferior
3. Si es > busco en la mitad superior

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Buscar el valor 56 en el vector



¿Cómo es 56 con respecto a  $v[\text{medio}]$ ?

1. Si es = terminé
2. Si es < busco en la mitad inferior
3. Si es > busco en la mitad superior



# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Buscar el valor 56 en el vector

v	10	34	56	123	234	265	397	400	405
---	----	----	----	-----	-----	-----	-----	-----	-----

Observemos que :

1. La primera vez se trabaja con el vector completo para determinar el punto medio
2. La siguiente vez, el vector se reduce a la mitad
3. La siguiente vez, el vector se reduce a la mitad de la mitad

*¿Cómo se calcula el medio?*

*¿Cómo se calcula la primera mitad?*

*¿Cómo se calcula la segunda mitad?*

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

**Buscar** (vector, datoABuscar)

si el vector “no tiene elementos” entonces

No lo encontré y termino la búsqueda

sino

Determinar el punto medio del vector

Comparar **datoABuscar** con el contenido del punto medio

si coincide entonces

“Lo encontré”

sino

si **datoABuscar** < contenido del punto medio entonces

**Buscar** (1era mitad del vector, datoABuscar)

sino

**Buscar** (2da mitad del vector, datoABuscar)

3) Existen 2 casos que se resuelven de manera directa (casos base):

- a) Cuando el vector “no contiene elementos”
- b) Cuando encuentro el datoABuscar

2) En cada llamada, el tamaño del vector se reduce a la mitad.

1) El módulo realiza invocaciones a si mismo (Caso recursivo)

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

**Buscar** (vector, datoABuscar)

si el vector “no tiene elementos” entonces

    No lo encontré y termino la búsqueda

sino

    Determinar el punto medio del vector

    Comparar **datoABuscar** con el contenido del punto medio

    si coincide entonces

        “Lo encontré”

    sino

        si **datoABuscar** < contenido del punto medio entonces

**Buscar** (1era mitad del vector, datoABuscar)

        sino

**Buscar** (2da mitad del vector, datoABuscar)

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

**Buscar** (vector, datoABuscar)

**si** el vector “no tiene elementos” **entonces**

No lo encontré y termino la búsqueda

**sino**

Determinar el punto medio del vector

Comparar **datoABuscar** con el contenido del punto medio

**si** coincide **entonces**

“Lo encontré”

**sino**

**si** **datoABuscar** < contenido del punto medio **entonces**

**Buscar** (1era mitad del vector, datoABuscar)

**sino**

**Buscar** (2da mitad del vector, datoABuscar)

# Recursión

Implementación del  
caso recursivo  
(autoinvocación)

## Ejemplo: Búsqueda dicotómica en un vector

**Buscar** (vector, datoABuscar)

si el vector “no tiene elementos” entonces

No lo encontré y termino la búsqueda

sino

Determinar el punto medio del vector

Comparar **datoABuscar** con el contenido del punto medio

si coincide entonces

“Lo encontré”

sino

si **datoABuscar** < contenido del punto medio entonces

**Buscar** (1era mitad del vector, datoABuscar)

sino

**Buscar** (2da mitad del vector, datoABuscar)

# Recursión

# 2<sup>3</sup>

## Ejemplo: Potencia de un número

```
program CalculoDePotencia;  
var base, exponente, potencia, i: integer;  
begin  
  base := 2;  
  exponente := 3;  
  potencia := 1;  
  for i := 1 to exponente do  
    potencia := potencia * base;  
  writeln(potencia),  
  readln;  
end.
```

### Planteo de solución recursiva. Tener en cuenta:

1. ¿Cómo defino el problema en términos de problemas más simples del mismo tipo?
2. ¿Cómo achico el problema en cada llamado recursivo?
3. ¿Qué instancia/s del problema son caso/s base?

### Cálculo de 2<sup>3</sup>:

$$2^3 = 2 * 2^2 = 2 * \underbrace{2 * 2^1}_{2^2} = 2 * 2 * \underbrace{2 * 2^0}_{2^1} = 2 * 2 * 2 * \underbrace{1}_{2^0} = 8$$

# Recursión

Ejemplo: Potencia de un número

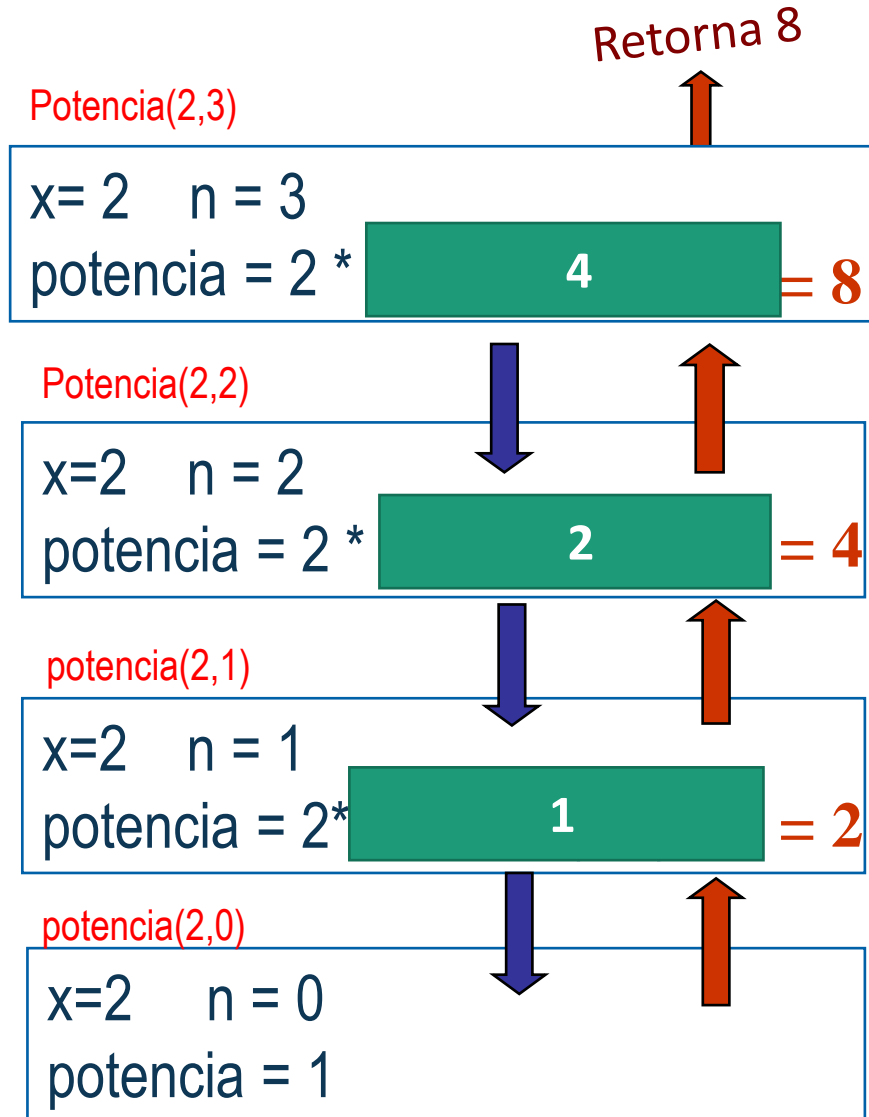
$$X^n = \begin{cases} 1 & \text{si } n = 0 \\ X * X^{n-1} & \text{si } n \geq 1 \end{cases}$$

```
Function potencia (x, n: integer): real;  
begin  
  if (n = 0) then  
    potencia := 1  
  else  
    potencia := x * potencia(x, n-1);  
  end;  
end;
```

# Recursión

## Ejemplo: Potencia de un número

```
program ejemplo;  
  
function potencia (x,n:integer): real;  
begin  
  if (n = 0) then  
    potencia := 1  
  else  
    potencia := x * potencia(x,n-1);  
  end;  
  
var  
  x,n:integer;  
begin  
  read (x,n);  
  write (potencia(x,n));  
end.
```







# Actividades en Máquina

## ACTIVIDAD 1

$$X^n = \begin{cases} 1 & \text{si } n = 0 \\ X * X^{n-1} & \text{si } n \geq 1 \end{cases}$$

Crear el programa **CalculoDePotencia.pas**

a) Implementar en el programa **CalculoDePotencia**, la función **potencia1**

```
Function potencia1 (x,n: integer): real;  
begin  
    potencia1 := x * potencia1(x,n-1);  
end;
```

b) Invocar a la función **potencia1** para calcular  $5^3$  .

c) Compilar y ejecutar. ¿Qué ocurre? ¿Por qué?



# Actividades en Máquina

## ACTIVIDAD 2

a) Implementar en el programa **CalculoDePotencia**, la función **potencia2**

```
Function potencia2 (x,n: integer): real;  
begin  
  if (n = 0) then  
    potencia2:= 1  
  else  
    potencia2 := x * potencia2(x,n);  
end;
```

b) Invocar a la función potencia2 para calcular  $5^3$ .

c) Compilar y ejecutar. ¿Qué ocurre? ¿Por qué?



# Actividades en Máquina

## ACTIVIDAD 3

Descargar el programa **Recursion.pas**

- a) Repase el procedimiento **digitoMaximoRec**
  - ¿Cuál es el caso base?
  - ¿Cómo se acerca al caso base?
- b) Compile, ejecute y compruebe el resultado.



# Actividades en Máquina

## ACTIVIDAD 4

Utilizando el programa **Recursion.pas** realice las siguientes actividades:

a) Modificar el procedimiento **digitoMaximoRec**. Debe colocarse la instrucción **writeln ('max: ', max);** *después* de la autoinvocación.

b) Responder:

¿Qué valor se muestra antes de finalizar cada instancia recursiva?

¿Qué valor se muestra en el programa principal?



# Actividades en Máquina

## ACTIVIDAD 5

Utilizando el programa **Recursion.pas** realice las siguientes actividades:

a) Modificar el procedimiento **digitoMaximoRec**. Debe colocarse la instrucción **writeln ('max: ', max);** antes de la autoinvocación.

b) Responder:

¿Qué valor se muestra antes de cada llamada recursiva? ¿Por qué?

¿Qué valores se imprimen si el parámetro max es pasado por valor? ¿Qué imprime en el programa? ¿Funciona?



# Actividades en Máquina

## ACTIVIDAD 6

Si el nro es: 5236  
**ImprimirDigitos1**

6  
3  
2  
5

Si el nro es: 5236  
**ImprimirDigitos2**

5  
2  
3  
6

En el programa **Recursion.pas**

- a) Implementar el procedimiento recursivo **ImprimirDigitos1** que imprime los dígitos de un número dado, empezando por la unidad.
- b) Implementar el procedimiento recursivo **ImprimirDigitos2** que imprime los dígitos de un número dado, finalizando con la unidad.

Nota: el planteo de la solución es similar a la del procedure digitoMaximoRec



# Actividades en Máquina

## ACTIVIDAD 7

Crear el programa **ListaConRecursion.pas** que:

- a) Genere una lista de números enteros y muestre los valores guardados.
- b) Invoque a un módulo recursivo **ImprimirEnOrden** que imprima los valores contenidos en la lista en el orden en que se guardaron.
- c) Invoque a un módulo recursivo **ImprimirOrdenInverso** que imprima los valores contenidos en la lista desde el último dato al primero.